Amendments to the Claims:

This listing of claims will replace all prior versions, and listings, of claims in the application:

Listing of Claims:

1. (Original) A method in a data processing system for managing an application's persistent data across multiple different release versions of said application, said method comprising the steps of:

defining a format for a memory area wherein said persistent data will be stored according to a first release version of said application;

organizing said format to permit said application running at different release versions to access said memory area; and

accessing said memory area by said application that is running at a second release version utilizing said format.

2. (Original) The method according to claim 1, further comprising the steps of:

said step of defining a format further comprising the step of logically dividing said memory area into individually accessible sections; and

maintaining a template of a layout for each one of said sections for each one of said release versions.

3. (Original) The method according to claim 1, further comprising the steps of:

storing first data in said memory area for said first release version and storing second data in said memory area for said second release version; and

retrieving only said second data from said memory area using said application that is running at a second release version and disregarding said first data.

4. (Original) The method according to claim 1, further comprising the steps of:

storing in said memory area data that is specific to each one of said release versions;

retrieving from said memory area, by said application that is running at said second release version, data that is specific to only said second release version; and

disregarding, by said application that is running at said second release version, data that is specific to ones of said release versions other than said second release version.

5. (Original) A method in a data processing system for managing an application's persistent data, stored within a memory area, across multiple different release versions of said application, said method comprising the steps of:

logically dividing said memory area into individually accessible sections;

defining a layout for each one of said sections for storing data for each one of said release versions;

creating a template of said layout for each one of said sections for each one of said release versions; and

accessing a current release version of said memory area by said application that is running at a particular release version using a template of said layout for each section created for said particular release version, said current release version being different from said particular release version.

6. (Original) The method according to claim 5, further comprising the steps of:

numbering each one of said sections sequentially;

determining which ones of said sections are included in each one of said release versions; and

determining which ones of said sections are included in said current release version of said memory area.

7. (Original) The method according to claim 5, further comprising the steps of:

setting aside a length field in each one of said sections in which to store a length of said section;

setting aside a field in said memory area in which to store an identification of all sections included in said memory area for each one of said release versions; and

setting aside a length field in each one of said sections in which to store a length for each variable length complex data structure included in said section.

8. (Original) The method according to claim 5, further comprising the steps of:

for each one of said sections, generating an offset into each one of said sections of said memory area utilizing said template of said layout for each section created for said particular release version, said offsets being used to determine where each one of said sections is located within said memory area.

9. (Original) The method according to claim 8, further comprising the steps of:

adjusting said offsets when said current release version of said memory area includes a section that is not expected by said application that is running at said particular release version.

10. (Original) The method according to claim 8, further comprising the steps of:

parsing each one of said sections of said memory area by said application;

determining during said parsing whether each one of said sections of said memory area is an expected length; and

in response to a determination that one of said sections is not said expected length, adjusting an offset for one of said sections and adjusting an offset for all of said sections that are located after said one of said sections.

11. (Original) A data processing system for managing an application's persistent data across multiple different release versions of said application, said system comprising:

a format for a memory area wherein said persistent data will be stored according to a first release version of said application;

said format being organized to permit said application running at different release versions to access said memory area; and

said application that is running at a second release version accessing said memory area utilizing said format.

12. (Original) The system according to claim 11, further comprising:

said memory area being logically divided into individually accessible sections; and

a template of a layout for each one of said sections for each one of said release versions.

13. (Original) The system according to claim 11, further comprising:

first data being stored in said memory area for said first release version and second data being stored in said memory area for said second release version; and

said application that is running at a second release version retrieving only said second data from said memory area and disregarding said first data.

14. (Original) The system according to claim 11, further comprising:

data being stored in said memory area that is specific to each one of said release versions;

said application that is running at said second release version retrieving from said memory area data that is specific to only said second release version; and

said application that is running at said second release version disregarding data that is specific to ones of said release versions other than said second release version.

15. (Original) A data processing system for managing an application's persistent data, stored within a memory area, across multiple different release versions of said application, said system comprising:

said memory area being logically divided into individually accessible sections;

a layout for each one of said sections for storing data for each one of said release versions;

a template of said layout for each one of said sections for each one of said release versions; and

said application that is running at a particular release version accessing a current release version of said memory area using a template of said layout for each section created for said particular release version, said current release version being different from said particular release version.

16. (Original) The system according to claim 15, further comprising:

each one of said sections being numbered sequentially;

said system including a CPU executing code for determining which ones of said sections are included in each one of said release versions; and

said CPU executing code for determining which ones of said sections are included in said current release version of said memory area.

17. (Original) The system according to claim 15, further comprising:

a length field in each one of said sections for storing a length of said section;

a field in said memory area for storing an identification of all sections included in said memory area for each one of said release versions; and

a length field in each one of said sections for storing a length of each variable length complex data structure included in said section.

18. (Original) The system according to claim 15, further comprising:

for each one of said sections, said system including a CPU executing code for generating an offset into each one of said sections of said memory area utilizing said template of said layout for each section created for said particular release version, said offsets being used to determine where each one of said sections is located within said memory area.

19. (Original) The system according to claim 18, further comprising:

said CPU executing code for adjusting said offsets when said current release version of said memory area includes a section that is not expected by said application that is running at said particular release version.

20. (Original) The system according to claim 18, further comprising:

said system including a CPU executing code for parsing each one of said sections of said memory area by said application;

said system including a CPU executing code for determining during said parsing whether each one of said sections of said memory area is an expected length; and

in response to a determination that one of said sections is not said expected length, said system including a CPU executing code for adjusting an offset for one of said sections and adjusting an offset for all of said sections that are located after said one of said sections.

21. (Original) A computer program product in a data processing system for managing an application's persistent data, stored within a memory area, across multiple different release versions of said application, said product comprising:

instruction means for logically dividing said memory area into individually accessible sections;

instruction means for defining a layout for each one of said sections for storing data for each one of said release versions;

instruction means for creating a template of said layout for each one of said sections for each one of said release versions; and

instruction means for accessing a current release version of said memory area by said application that is running at a particular release version using a template of said layout for each section created for said particular release version, said current release version being different from said particular release version.

22. (Original) The product according to claim 21, further comprising:

instruction means for numbering each one of said sections sequentially;

instruction means for determining which ones of said sections are included in each one of said release versions; and

instruction means for determining which ones of said sections are included in said current release version of said memory area.

23. (Original) The product according to claim 21, further comprising:

instruction means for setting aside a length field in each one of said sections in which to store a length of said section;

instruction means for setting aside a field in said memory area in which to store an identification of all sections included in said memory area for each one of said release versions; and

instruction means for setting aside a length field in each one of said sections in which to store a length for each variable length complex data structure included in said section.

24. (Original) The product according to claim 21, further comprising:

for each one of said sections, instruction means for generating an offset into each one of said sections of said memory area utilizing said template of said layout for each section created for said particular release version, said offsets being used to determine where each one of said sections is located within said memory area.

25. (Original) The product according to claim 24, further comprising:

instruction means for adjusting said offsets when said current release version of said memory area includes a section that is not expected by said application that is running at said particular release version.

26. (Original) The product according to claim 24, further comprising:

instruction means for parsing each one of said sections of said memory area by said application;

instruction means for determining during said parsing whether each one of said sections of said memory area is an expected length; and

in response to a determination that one of said sections is not said expected length, instruction means for adjusting an offset for one of said sections and adjusting an offset for all of said sections that are located after said one of said sections.